# WORK REPORT

June 03,2024 - Aug 2,2024

Name: **Sanyam Gupta**
ID: **sanyam.1**

# Introduction

## Speaker Verification vs Speaker Identification

- **Speaker Verification** : Speaker verification aims to verify whether a claimed speaker (user) matches their enrolled voiceprint (template).
    - Process
        - The user provides a voice sample.
        - The system compares it to the stored template.
        - Decision: Accept (genuine) or reject (impostor).
- **Speaker Identification** : Speaker identification identifies an unknown speaker from a set of known speakers.
    - Process
        - The system compares the voice sample to a database.
        - Decision: Which speaker from the database matches?

.

# Types of SV systems

Modern speaker verification (SV) systems can be categorized into two main structures:.

## Cascaded Structure

- Comprises a front-end and a backend.
- Utilizes an embedding model (e.g., i-vector extractor or deep embedding network) to produce speaker embeddings.
- A backend classifier computes verification scores based on these embeddings

## End-to-End Structure

- Directly outputs verification scores or decisions.
- No intermediate embedding step; scores are computed directly from the input data

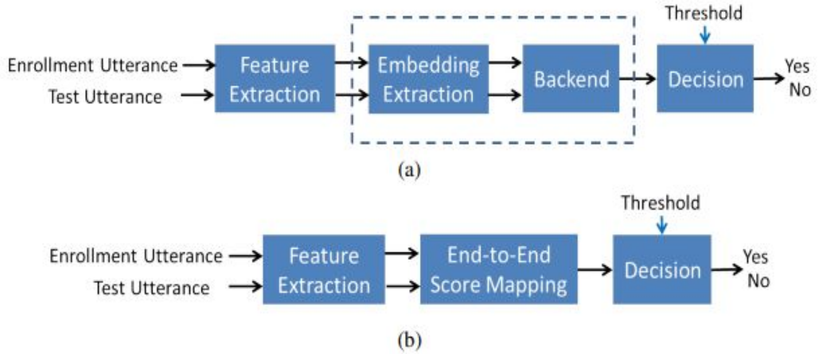



FIGURE 1: (a) A "Front-end + Backend" cascaded SV system and (b) an end-to-end SV system.

*DOI 10.1109/ACCESS.2022.3206541*

# Text Independent v/s Text Dependent Speaker Verification

- **TI-SV (Text-Independent Speaker Verification):**
    - **Content Freedom:** No lexical constraints; users speak naturally
    - **Training Data:** Trained on long utterances to handle phonetic variability.
    - **Advantages:**
        - Suitable for diverse speech content.
        - Passive recognition; no predefined phrases
    - **Disadvantages:**
        - Longer training duration.
        - Handling session variability.
- **TD-SV (Text-Dependent Speaker Verification):**
    - **Lexical Constraints:** Constrained to specific words or phrases.
    - **Advantages:**
        - Excels in short-duration scenarios.
        - Quick response.
    - **Disadvantages:**
        - Requires in-domain data (expensive).

# Text-Independent Speaker Verification (TI-SV)

Most modern TISV Systems use Cascaded Structure.

## Components

- **Front-End:**
  - Extracts speaker characteristics from audio data.
  - Common front-ends include:
    - **i-vector embedding:** Represents speakers as fixed-length vectors.
    - **x-vector embedding:** Utilizes time delay neural networks (TDNNs) for frame-level features.
    - **Deep speaker embedding:** Employs neural networks to create a speaker-embedding space.
- **Backend:**
  - Scores the similarity between speaker embeddings.
  - Common backends include:
    - **Cosine similarity measure:** Compares the cosine angle between embeddings.
    - **Probabilistic linear discriminant analysis (PLDA):** A statistical model for scoring.

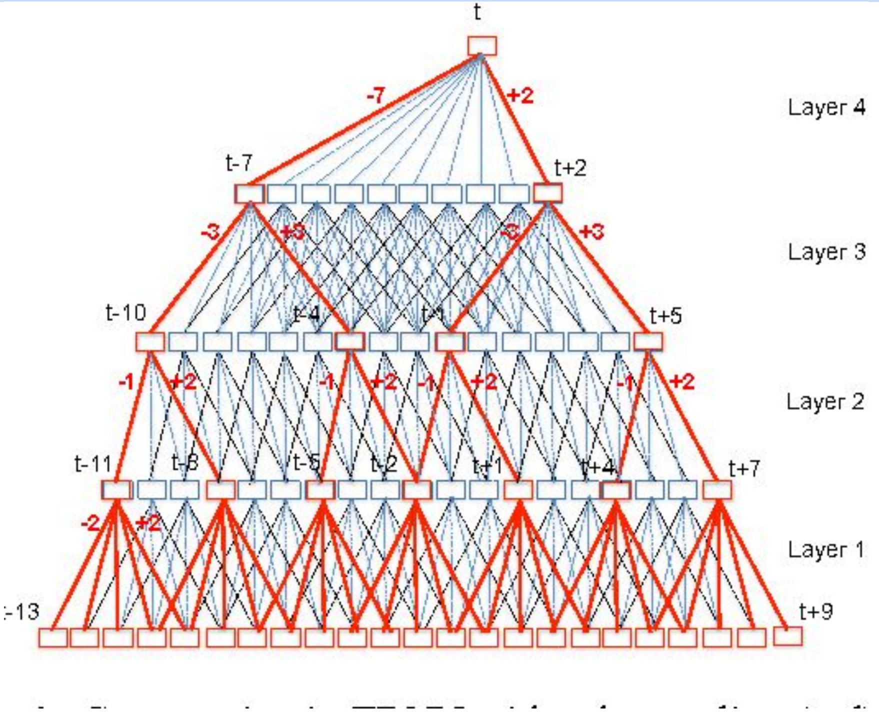
# Speaker Embedding Extraction

## Architecture:

- **Frame-level subnetwork:**
  - Typically based on convolutional neural networks (CNNs).
  - Classical example is x-vector which utilizes time delay neural networks (TDNNs) for frame-level feature extraction.
  - Later advancements include ResNets, DenseNets, and Res2Nets to improve modeling of spectral-temporal relationships.
- **Pooling layer:**
  - Aggregates frame-level features into utterance-level embeddings.
  - Methods: statistics pooling, multi-head attentive pooling, NetVLAD-based pooling, short-time spectral pooling.
- **Utterance-level subnetwork:**
  - Extracts speaker embeddings from the FC layer output.
  - Training losses (besides softmax):
    - Additive margin softmax (AM-Softmax).
    - Additive angular margin softmax (AAM-Softmax).

# TDNN Basics

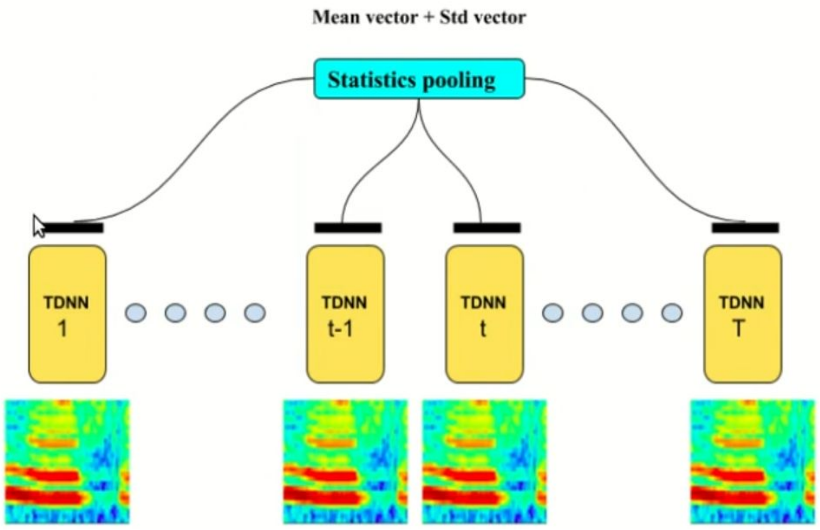
- TDNN captures long term temporal correlations between speech frames
- They are faster compared to LSTM.
- They learn local correlations between speech frames.
- TDNN use context windowing to get better accuracy.
- MFCC vector of one frame is the input provided(one box).
- TDNN process them with fixed local temporal context.
- The context width increases as we go to upper layers.




doi: 10.21437/Interspeech.2015-647

# Statistics Pooling

- Statistics Pooling layer aggregates frame level features into a representation of whole utterance.
- Uses mean and standard deviation of all the frame level feature vector.



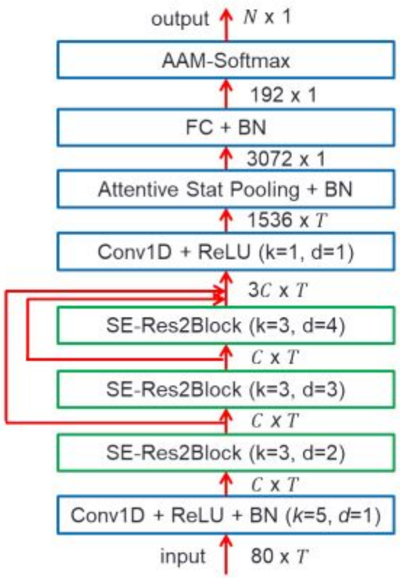

*doi: 10.21437/Interspeech.2015-647*

# ECAPA TDNN System

Emphasized Channel Attention, Propagation and Aggregation in TDNN (ECAPA-TDNN) System follows the framework of a x-vector extractor. It achieved state of the art performance on VoxCeleb1 dataset.

## Key Differences

- Use of Res2Net Blocks:
  - They enhance information propagation by allowing multi-layer feature aggregation.
- Specialized pooling layer:
  - This layer adapts to both channel characteristics (e.g., which channels are discriminative) and content characteristics (e.g., the specific features in the input).
- AAM-Softmax loss:
  - Instead of vanilla softmax loss, AAM-Softmax loss is used to improve discriminative power of the system.




(a)

DOI 10.1109/ACCESS.2022.3206541

# Dataset

Github Repository used : [https://github.com/TaoRuijie/ECAPA-TDNN](https://github.com/TaoRuijie/ECAPA-TDNN)

## Datasets used for Training

- VoxCeleb2 training set
- MUSAN dataset
- RIR dataset

.

## Datasets used for Testing

- VoxCeleb1 test set for Vox1_O
- VoxCeleb1 train set for Vox1_E

.

**Verification split**

| | dev | test |
|---|---|---|
| # of speakers | 1,211 | 40 |
| # of videos | 21,819 | 677 |
| # of utterances | 148,642 | 4,874 |

Fig: VoxCeleb1 split

| | dev | test |
|---|---|---|
| # of speakers | 5,994 | 118 |
| # of videos | 145,569 | 4,911 |
| # of utterances | 1,092,009 | 36,237 |

Fig: VoxCeleb2 split

# Data format for training & DataLoader

## About the Data format

- File Format: .wav
- Length: Random 2 or 3 seconds from each utterance(fixed length)(If duration is not enough, require padding)
- Content: speech only

## Dataloader

- Read the official training list, map speaker ID to class ID

```
File list: ['id00012/C_FAL9gv8bo/00026.wav',
            'id00012/C_FAL9gv8bo/00026.wav',
            'id00015/HG2AS_DV241/00001.wav',
                     .....]
Label list: [0,0,1,....]
```

```
id00012 id00012/21Uxsk56VDQ/00015.wav
id00012 id00012/2DLq_Kkclr8/00016.wav
id00012 id00012/2DLq_Kkclr8/00017.wav
id00012 id00012/2DLq_Kkclr8/00018.wav
id00012 id00012/73OrGYvy4ng/00019.wav
id00012 id00012/C_FAL9gv8bo/00020.wav
id00012 id00012/C_FAL9gv8bo/00021.wav
id00012 id00012/C_FAL9gv8bo/00022.wav
id00012 id00012/C_FAL9gv8bo/00023.wav
id00012 id00012/C_FAL9gv8bo/00024.wav
id00012 id00012/C_FAL9gv8bo/00025.wav
id00012 id00012/C_FAL9gv8bo/00026.wav
id00012 id00012/C_FAL9gv8bo/00027.wav
```

Training list

# Feature Extraction

- **Feature Extraction(FBank) :**
  - Pre-Emphasis: Amplify the high frequency.
  - Frame blocking and windowing: Split the signal into short time frames.
  - Fourier Transform: Do an N-point FFT on each frame (Short-Time Fourier-Transform (STFT)) to calculate the frequency spectrum
  - Filter Bank:Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank
  - Coding: Using feature extraction toolkit of various machine learning libraries.

```
self.torchfbank = torch.nn.Sequential(
    PreEmphasis(),
    torchaudio.transforms.MelSpectrogram(sample_rate=16000, n_fft=512, win_length=400, hop_length=160, \
                                          f_min = 20, f_max = 7600, window_fn=torch.hamming_window, n_mels=80),
    )
```

# Evaluation Pipeline

- **Evaluation Pipeline :**
  - **Input:** Utterance A and utterance B (Raw wav files, without data augmentation)
  - **Then:** Get speaker embedding from A and B
  - **Output:** The similarity score between these two utterances. Compare with threshold to get final output.

1 is positive pair
0 is negative pair
These speakers are not used during training

```
1 id10270/x6uYqmx31kE/00001.wav id10270/8jEAjG6SegY/00008.wav
0 id10270/x6uYqmx31kE/00001.wav id10300/ize_eiCFEg0/00003.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/GWXujl-xAVM/00017.wav
0 id10270/x6uYqmx31kE/00001.wav id10273/0OCW1HUxZyg/00001.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/8jEAjG6SegY/00022.wav
0 id10270/x6uYqmx31kE/00001.wav id10284/Uzxv7Axh3Z8/00001.wav
1 id10270/x6uYqmx31kE/00001.wav id10270/GWXujl-xAVM/00033.wav
0 id10270/x6uYqmx31kE/00001.wav id10284/7yx9A0yzLYk/00029.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/5r0dWxy17C8/00026.wav
0 id10270/x6uYqmx31kE/00002.wav id10285/m-uILToQ9ss/00009.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXujl-xAVM/00035.wav
0 id10270/x6uYqmx31kE/00002.wav id10306/uzt36PBzT2w/00001.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXujl-xAVM/00038.wav
0 id10270/x6uYqmx31kE/00002.wav id10307/kp_GCjLq4qA/00004.wav
1 id10270/x6uYqmx31kE/00002.wav id10270/GWXujl-xAVM/00033.wav
```

Fig: Evaluation File Format

# Implementing the model

## Major Challenges:

- **Dataset size:**
  - VoxCeleb dataset contains over 150000 utterances of 1000+ celebrities extracted from Youtube.
  - I did not have storage and CPU bandwidth to download and train the model on VoxCeleb+augmentation datasets.
  - Hence i downloaded librispeech dataset which contains 28,539 training utterances and 2,620 test utterances.
- **Modifying the script**
  - Librispeech dataset has different folder structure and extension compared to voxceleb dataset. Hence i modified relevant parts of the script.
  - Edited out the code relevant for data augmentation.
  - Changed the code of DataLoader so that it correctly loads the audio files given different structure of the librispeech dataset.

- **No Training/Testing list:**
    - Librispeech dataset did not have an official training and testing list like VoxCeleb dataset which is necessary for code to run.
    - Used a python script to create testing pairs and attached relevant labels to them based on speaker ID.
    - Edited the list so that it contains about 600 testing pairs.

```
test-clean\6930\76324\6930-76324-0028.flac,test-clean\6930\81414\6930-81414-0024.flac,1
test-clean\7729\102255\7729-102255-0035.flac,test-clean\7729\102255\7729-102255-0044.flac,1
test-clean\3575\170457\3575-170457-0017.flac,test-clean\3575\170457\3575-170457-0032.flac,1
test-clean\260\123288\260-123288-0021.flac,test-clean\260\123440\260-123440-0020.flac,1
test-clean\7127\75947\7127-75947-0010.flac,test-clean\8455\210777\8455-210777-0004.flac,0
test-clean\7021\79740\7021-79740-0002.flac,test-clean\7021\85628\7021-85628-0009.flac,1
test-clean\4446\2271\4446-2271-0004.flac,test-clean\4446\2273\4446-2273-0005.flac,1
test-clean\1320\122617\1320-122617-0000.flac,test-clean\1580\141083\1580-141083-0047.flac,0
test-clean\3729\6852\3729-6852-0023.flac,test-clean\237\134500\237-134500-0036.flac,0
test-clean\5639\40744\5639-40744-0005.flac,test-clean\5639\40744\5639-40744-0029.flac,1
test-clean\5683\32866\5683-32866-0006.flac,test-clean\8555\292519\8555-292519-0001.flac,0
test-clean\121\121726\121-121726-0004.flac,test-clean\121\127105\121-127105-0004.flac,1
test-clean\237\134493\237-134493-0012.flac,test-clean\1284\1181\1284-1181-0014.flac,0
test-clean\6930\81414\6930-81414-0019.flac,test-clean\8555\284447\8555-284447-0001.flac,0
```

# Results on Pre-Trained models

Found two Github repositories which provided pretrained model parameters. Hence I evaluated the testing dataset of librispeech against those two.

## Model Trained on VoxCeleb Dataset

- Github Repo:https://github.com/TaoRuijie/ECAPA-TDNN
- Did 3 runs with different testing list each time
- Average EER: 2.27%
- minDCF: 0.045%

## Model Trained on Japanese Speaker Dataset

- Github Repo: https://github.com/k-washi/speaker-emb-ja-ecapa-tdnn
- Trained on japanese speaker dataset with 4096 speakers.
- Did 3 runs with different testing list
- Average EER: 12.87%

# Results on Custom Training

## Changes Made

- Reduced number of training files to 1k.
- Removed Data Augmentation and Changed Dataloader to correctly load dataset.
- Used to python Script to create both training and testing lists.

## Results

- Used Clean audio files for testing, as training was not done using data augmentation.
- Used around 600 testing pairs.
- Got error rate of 4.2%

.

# THANK YOU